# Arduino Wine Thermostat

masterfelix312, Kozani 21/08/2017

## Project Abstract

The purpose of this project is to automate the procedure of controling the wine fermentation temperature. It is only demanded to set a desired temperature and the device can cool down the wine to this one. The cooling is achieved by the tangential tap water flow on the outer surface of a stainless steel cylindrical tank that the wine is contained.

(a) Cylindrical fermentation tank.

(b) The grapes.

## Introduction

The wine making process is very sensitive to environment temperature, humidity and oxygen levels. The most important stage is the fermentation and the variable that can most affect the chemical reactions involved, at an open tank fermentation process, is the temperature. There is a specific temperature range ($15 - 25°$C) that allows the fermentation to take place and assuming that the ambient temperature is approximately $20°$C, our only concern is to keep the temperature below $25°$C. The fermentation produces heat so we need to dissipate that heat away from the wine to keep the temperature to a certain value.

The manual way to control the temperature is by measuring its value with a thermometer submersed to the liquid and if it is over the desired temperature, we allow a flow of water touching the tank. So, the way to automate this function is by constantly measuring the temperature with a sensor, that is always submersed to the center of the cylindrical tank, and control electronically the flow of water according to the sensor readings.

(a) 4 digits 7-segment display


(b) Waterproof temperature sensor DS18B20

(c) Supply List

| Item | Quantity | Price |
|---|---|---|
| 12Volt Solenoid Valve | 1 | 10.05 € |
| Waterproof DS18B20 Temperature Sensor | 1 | 6.99 € |
| Funduino Uno (Arduino Compatible) | 1 | 9.60 € |
| Project Box 170x84x36mm | 1 | 2.01 € |
| Prototyping Board 70x90cm | 1 | 1.21 € |
| Power Sypply 12V 1.5A | 1 | 5.00 € |
| Tactile Buttons | 2 | 0.90 € |
| 4 Digit 7 segment display | 1 | 0.90 € |
| 8bit Shift Register | 1 | 0.50€ |
| Relay 5V DC | 1 | 0.60 € |
| Resistor 220Ω | 3 | 0.01 € |
| Resistor 10kΩ | 2 | 0.01€ |
| Resistor 4.7kΩ | 1 | 0.01€ |
| Pin Header 1x8 Male 2.54mm | 2 | 0.20 € |
| Pin Header 1x4 Male 2.54mm | 1 | 0.20 € |
| Screw Terminal 2P | 2 | 0.20 € |
| Electrolytic Capacitor 25V 4.7uF | 1 | 0.05 € |
| DIP Socket 16pin | 1 | 0.06 € |
| Total | | 39.83 € |

# Description of Arduino solution

## Hardware

## Specification

- Initially we place the temperature sensor inside the wine and thus by reading the temperature with arduino, we know the wine temperature at every moment (or time intervals that we define). We can also depict this reading with a 7-segment LED display.
- Using this kind of display and two push-down buttons we can offer two functions. First, the reading of the real time temperature and, second, to adjust the desired temperature pressing these two buttons. The display shows the adjustable desired temperature only when we press a button and for 1-2 seconds after this action. The resolution of $0.5°$C is considered sufficient for our purpose.
- When the real temperature does not coincide with the desired/set temperature a control signal of 5 Volts (digital output of an arduino pin) enables a relay that connects 12 Volts to the electrovalve in order to allow the water flow. There must be taken into consideration that a tolerance to this temperature difference is needed or alternatively a proper time delay

- The project consists of the arduino board, the board that hosts the other electronic parts, the sensor, the electrovalve and the power supply.

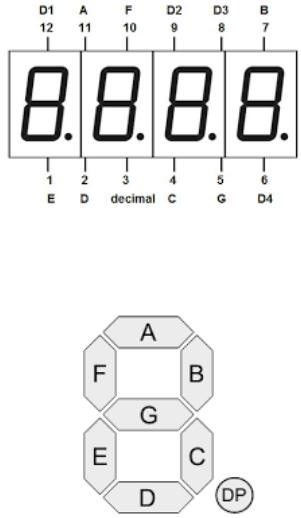
## Subsystems

This project consists of some basic subsystems that they can be impelemented and tested separately and then they can be combined to the final product.

### Display of a number

We choose to implement this display by minimazing the pins required from the microcontroller. In order to achieve that we use

1. A common anode **4 digits 7 segment display**. The concept is to connect every D# with an Arduino output pin to select successively the digits that we want to display. The selection is made with a HIGH output and to display a specific digit we define which segments will be lit with the next device we will need, the shift register

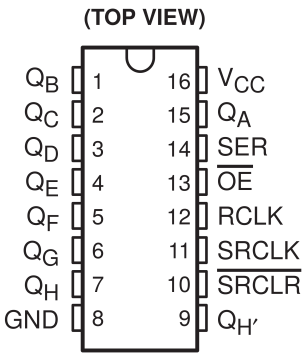



(a) Display pinout and a 7-segment digit

| | a | b | c | d | e | f | g | DP |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

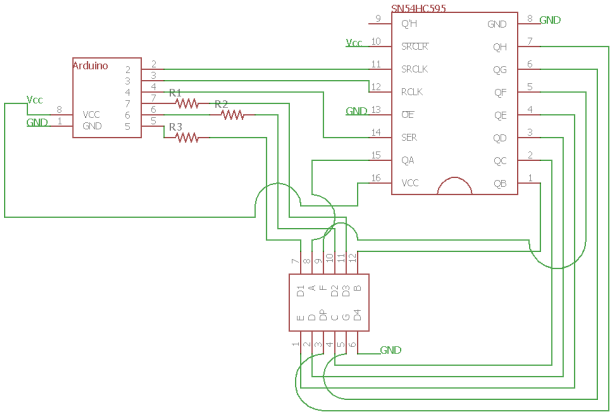
(b) Decimal to 7 segment

2. The **shift register** takes a serial input (pulsetrain of one byte) from one pin (SER) to output it parallel to the $Q_A - Q_H$ pins. In order for this byte to be inserted serially there is also a clock (SRCLK) needed. The positive edge of the clock defines the triggering moment of data sampling. The RCLK pin (or latch pin) must be triggered from LOW to HIGH in order to store the serial value to the register, which means that is transfered to the parallel output. For our circuit to function correctly we need to set the pins $\overline{\text{OE}}$ and $\overline{\text{SRCLR}}$ to LOW and HIGH, respectively.




**SN54HC595 . . . FK PACKAGE**

(a) Shift register pinout




(b) Shift register connections with Arduino and the display.

### Measuring temperature value

The sensor we use is DS18B20 that has 3 pins GND, VCC and DATA. This sensor can be used also to a cicuitry that is called parasite power and the sensor "steals" power from the data line to operate. In this circuitry, Vcc and GND are connected to 0 Volts and DATA is connected to an arduino pin that is dedicated to the communication with the sensor and through a 5k$\Omega$ resistor to 5 Volts. The communication is achieved by establishing a 1-wire interface using two libaries, OneWire and DallasTemperature.

### Enable electrovalve

This is the simplest subsystem, because we just use a relay to connect/disconnect the electrovalve with the 12 Volts from the power supply when a HIGH state (5 Volts) from an output pin is applied to the relay.

### Sense a pressed push-down button

Each push-button connects an input pin to 5 Volts and the pin is also connected to the ground through a 10k$\Omega$ resistor. So, when the button is not pressed the input pin is in LOW state, but when pressed is in HIGH state.

## Circuit diagram and perfboard manufacturing

In order to connect all those subsystems together and with the Arduino board, a handmade circuit board with the use of a perfboard was manufactured.

The manufacturing process of circuit connections on a perfboard demands skillful soldering and meticulous design. It is a more prone to errors procedure than etching a PCB, but it does not require chemicals etchers like Ferric Chloride.

The PCB Design software Eagle is a great help to the manufacture of the board and that is beacause we can design an exact sketch of the connections we need to make. The main purpose of the design is to contain the circuit to as small area as possible but also try to lay

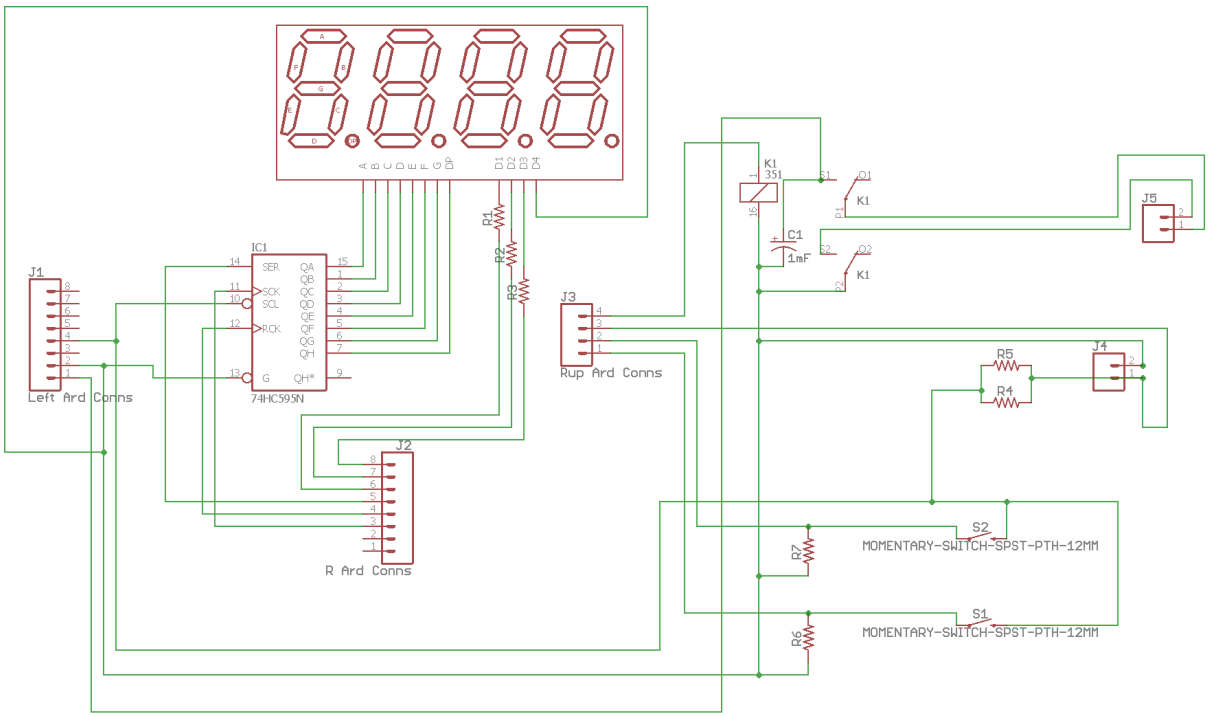
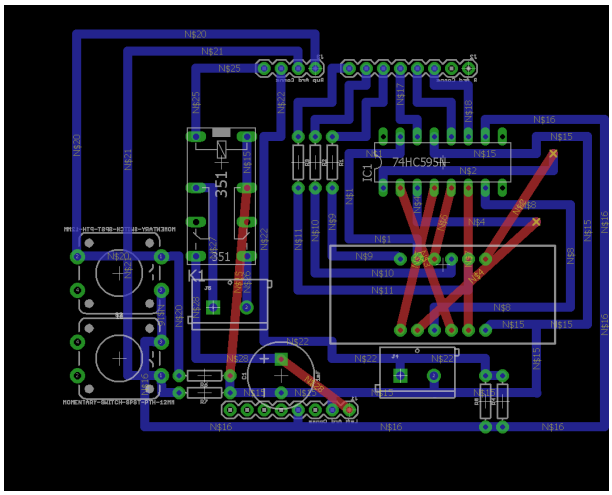
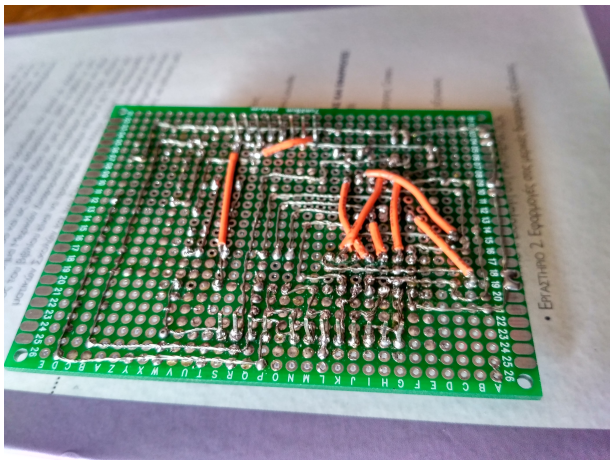



Figure 5: The circuitry of the project.



(a) The board schematic designed in Eagle.

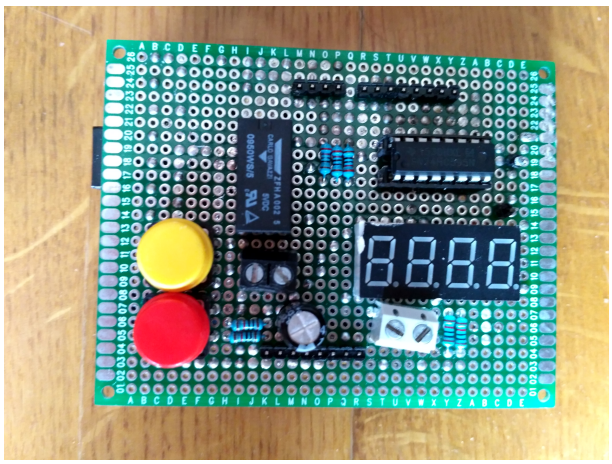


(b) The perfboard and the paths made on it.

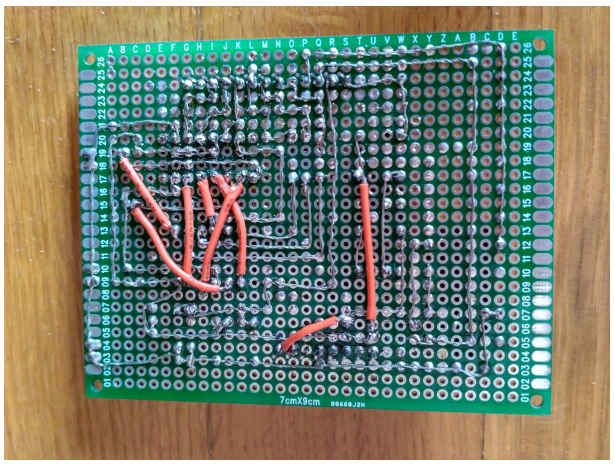
the connections at one plane. The later is not always possible so we have to make connections (red lines) over the rest paths.

The steps we need to take after the Eagle design are:

1. Firstly, we solder the components onto the perfboard according to the design
2. Then we make the paths with thin wire and solder also the points of the right angle turns of the paths. The beginning an the end of a path is soldered onto the pins of a component.
3. Lastly, we create with dielectric covered wire(orange wire in picture of perfboard) the connections that could not fit at the plane.



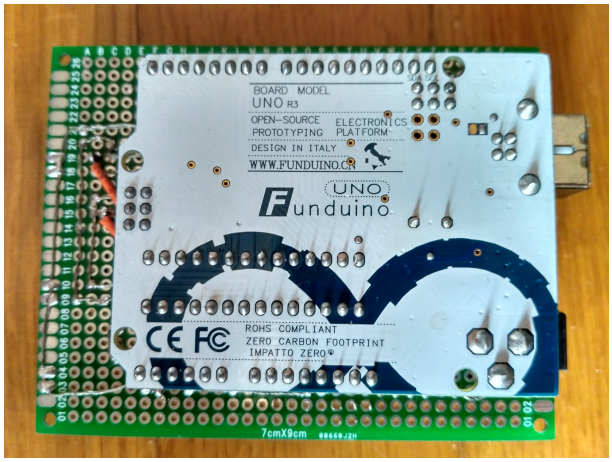
(a) The upper view of the perfboard, where the components exist.



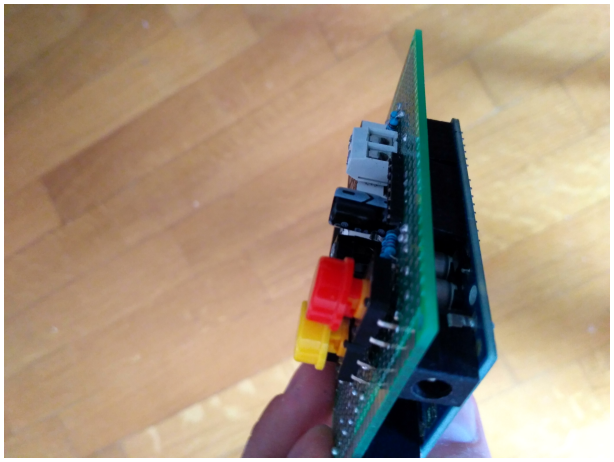
(b) Th e bootom view of the perfboard, where all the conducting paths exist.

## Assembly

As you can see in the images of the perfboard, at some points we have placed long pins. These pins are used to connect our circuit with the Arduino board in a double layer manner (sandwich).

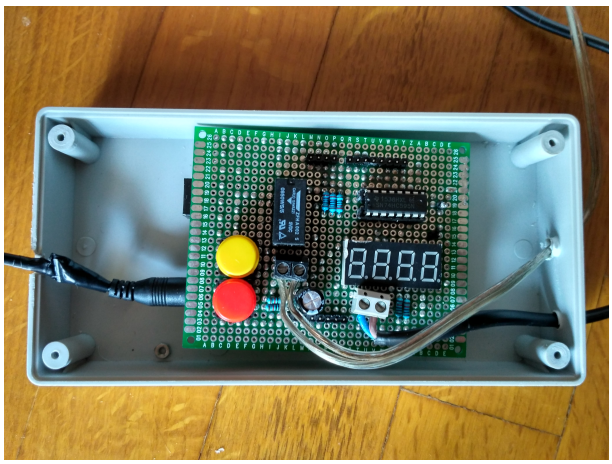



(a) The bottom view of the assembly.

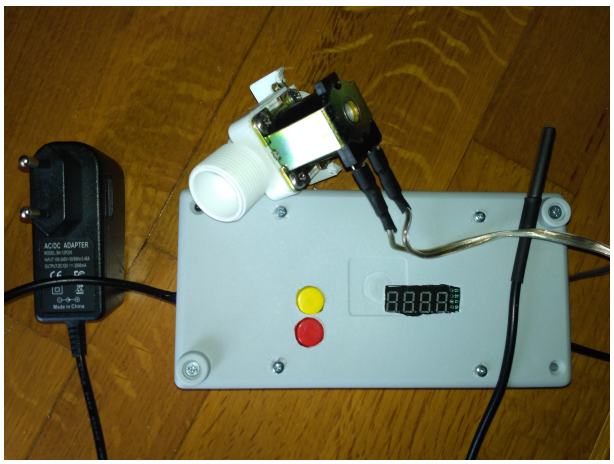


(b) The double layer.

The complete assembly of the project is achieved by housing the circuit along with the Arduino board inside a case that protects it and simultaneously gives access to the temperature readings and press-down buttons.



(a) The housing of the circuit board and Arduino.



(b) The complete project.

## Arduino Code

### Take code directly from *.ino file

```
/*Arduino-Wine-Thermostat Project
masterfelix312

This arduino code is made to toggle an electrovalve, which controls the flow of
cold water to the outer surface of a cylindrical steel tank to lower the
temperature of the wine that is contained in it. A temperature sensor is used to
monitor the temperature of the wine, two buttons to set it to the desired
temperature and a 7segment display to depict the readings.
*/

//These two libraries are essential for the easy extraction of temperature reading
//and communication between arduino microcontroller and the temperature
//sensor DS18B20

#include <OneWire.h>
#include <DallasTemperature.h>

//Data wire of sensor DS18B20 is plugged into port 10 on the Arduino
#define ONE_WIRE_BUS 10

// Setup a oneWire instance to communicate with any OneWire devices
//(not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

//Set the pins that correspond to each
//digit for the display
int Digits[3];
const int Dfirst=5;
const int Dlast=7;
```

```
33
//Set the pins that wil be used for the shift register operation and
35 //passing of values
const int ser=4;
37 const int latch=3;
const int clk=2;
39
//Set the pins that will be used as inputs of the push-down buttons
41 const int buttonUp=8;
const int buttonDown=9;
43
//Set the pin that toggles the electrovalve through a relay
45 const int relayEnable=11;

47 //The default values for the real-time temperature and the desired
//temperature
49 float temperatureCurr=25.5;
float temperatureSet=25.5;
51
//Initiate values used for renewing the display
53 unsigned long lastTime=0;
unsigned int count=0;
55


57 void setup() {

59   //Set as output all the digits enablers
   for (int i=0;i<3;i++){
61     Digits[i]=Dfirst+i;
     pinMode(Digits[i], OUTPUT);
63   }

65   //Start up the Dallas library
   sensors.begin();
67
   //Set as output the serial input pin ser,clk and latch
69   pinMode(ser,OUTPUT);
   pinMode(clk,OUTPUT);
71   pinMode(latch,OUTPUT);

73   //Initialize push down inputs
   pinMode(buttonUp, INPUT);
75   pinMode(buttonDown, INPUT);

77   //Set and initialize the relay output
   pinMode(relayEnable,OUTPUT);
79   digitalWrite(relayEnable,HIGH);
}
81


83 int int2dispint(int digit){
```

```
  //This function is a conversion of an integer [0-9]
  //to the byte that enables the appropriate LED
  //segments at a 7-segment display
    switch (digit){
      case 0:
        return int(B00000011);
      case 1:
        return int(B10011111);
      case 2:
        return int(B00100101);
      case 3:
        return int(B00001101);
      case 4:
        return int(B10011001);
      case 5:
        return int(B01001001);
      case 6:
        return int(B01000001);
      case 7:
        return int(B00011111);
      case 8:
        return int(B00000001);
      case 9:
        return int(B00001001);
    }
  }

void write_to_display(float digOut){
  //This function outputs to the 4digit 7-segment display
  //(using 2 integer digits and one decimal) the number digOut that
  //is its input

  //Compute the value of each digit for the float number digOut
  int digit[3];
  digit[0]=int(digOut/10);
  digit[1]=int(digOut)%10;
  digit[2]=int(digOut*10)%10;

  //Convert the 3 first digits to appropriate byte for display
  for (int i=0;i<3;i++){
    digit[i]=int2dispint(digit[i]);

    //Insert the decimal point to the second digit
    if (i==1) digit[i]-=1;
  }
  //Write/Depict to display each digit
  for (int i=0;i<3;i++){

    //Pass at the shift register the specific digit
    digitalWrite(latch,LOW);
    shiftOut(ser,clk,LSBFIRST,digit[i]);
```

```
    digitalWrite(latch,HIGH);

    //Pick and enable the right digit
    digitalWrite(Digits[i],HIGH);

    //Keep for some msecs the LEDs ON
    delay(6);

    //Disable the digit output
    digitalWrite(Digits[i],LOW);
  }
}


void loop() {

  //Read the temperature from the sensor every 500 loops
  count++;
  if (count%500==0){

    //Get the current temperature
    sensors.requestTemperatures();
    temperatureCurr=sensors.getTempCByIndex(0);

    //In case of an error
    if (temperatureCurr<-100){
      return;
    }
  }

  //Set the temperature using the increment buttons
  unsigned long int timeInterval=500;
  //Enter the condition when the button is pressed and at least
  // timeInterval has passed

  //Button up
  if (digitalRead(buttonUp) == HIGH && millis()-lastTime>=timeInterval) {

    //Keep track of the last time the temperature was changed
    lastTime=millis();

    //Check for digits overflow
    if (temperatureSet<99.5){

      //Increment +0.5
      temperatureSet+=0.5;
    }
  }

  //Button down
  if (digitalRead(buttonDown) == HIGH && millis()-lastTime>=timeInterval){
```

```
    //Keep track of the last time the temperature was changed
187     lastTime=millis();

189     //Check for digits underflow
        if (temperatureSet>0.5){
191
          //Decrement -0.5
193       temperatureSet-=0.5;
        }
195   }

197   //If in a short time interval the temperature set was changed then
      //show the temperature set, in every other case show the current
199   //sensor temperature
      if (millis()-lastTime<=3*timeInterval){
201     write_to_display(temperatureSet);
      }
203   else{
        write_to_display(temperatureCurr);
205   }

207   //Check the current sensor temperature and compare it with the set
      //temperature if there is a difference more than 0.5 degrees, then
209   // the relay output relayEnable is enabled.
      if (temperatureCurr-temperatureSet>=0.5){
211     digitalWrite(relayEnable,HIGH);
      }
213   else{
        digitalWrite(relayEnable,LOW);
215     }
}
```

wine_thermostat.ino